

# Trilinos Pre-Checkin Test Script

**James M. Willenbring**  
**Trilinos Framework & Tools Lead**  
**Applied Mathematics & Applications**  
**Sandia National Laboratories**

**Thanks to Ross Bartlett for contributing material for slides.**

- What is the purpose of the checkin test script and what does it do?
- I just want to use the script quickly and safely, and I don't care much about the details. Show me how to get started.
- What are some of the most useful advanced options?
- Where can I find all of the details?
- Not a comprehensive discussion of all script options.



## Pre-Checkin Test Script Purpose

---

Python script that performs safe pre-checkin testing

- Automatically figures out what Trilinos packages have been changed
- Automatically enables all downstream packages
- Configures, builds and runs tests
  - Built-in Configurations:
    - **MPI\_DEBUG** (Optimized compiler options, checked STL, etc.)
    - **SERIAL\_RELEASE** (varies other configure options)
  - Only enables Primary Stable Code!
  - Strong warning options (warnings as errors is a problem)
- Sends emails after each build case is finished
- Sends final email indicating if it is okay to commit or not
- Can automatically push changes (Recommended)
- Fully customizable (enabled packages, build cases, etc.)

- General git setup

```
$ eg config --global user.name "Joe A. User" # Set your username  
$ eg config --global user.email jauser@sandia.gov # Set your email address  
$ eg config --global color.ui true # Use colorized output
```

- Passwordless access to ssg

- <https://software.sandia.gov/trilinos/developer/ssh.html>
- Consider an empty passphrase (skip keychain portion of instructions)

- (Optional) create a dummy repository to push to if you would like to try a practice push without modifying the main repository. This allows a developer to try the checkin-test script without any risk.

```
$ mkdir ~/throwAway; cd ~/throwAway; mkdir base; cd base  
$ git clone --mirror software.sandia.gov:/space/git/Trilinos  
$ cd ..; mkdir test; cd test  
$ git clone ~/throwAway/base/Trilinos.git # passwordless access not required  
for test
```

- Make changes and **eg commit** (local commit) after verifying you want to commit your changes
  - TIP: if you aren't ready to commit, use **--local-do-all** instead of **--do-all** below, that will skip the **eg pull --rebase**
  - NOTE: A second eg pull --rebase is done before the final push
- If CMake can find BLAS, LAPACK, and MPI, the process is easy:  

```
$ cd $TRILINOS_HOME  
$ mkdir CHECKIN  
$ echo CHECKIN >> .git/info/exclude # ignore checkin directory for commit  
$ cd CHECKIN  
$ ../checkin-test.py --do-all
```
- Try the above first

- My test failed because ...
  - Look in <BUILD\_DIR>/configure.out for configure errors
  - Add necessary configure options to <BUILD\_DIR\_NAME>.config
    - NOTE: Do not enable and disable packages in .config files.
- When ready, instruct the script to push:  
`$ .../checkin-test.py --push # If everything passed, and --do-all was used`  
`$ .../checkin-test.py --do-all --push # If the tests have to be rerun, e.g. due to failures or using --local-do-all`



## checkin-test.py: Example Driver Script

---

- A driver script can be created to pass commonly used options to checkin-test.py, and provide version control.
- Example driver script (`checkin-test-<mymachine>.sh`):

```
#!/bin/bash
EXTRA_ARGS=$@
echo "-DBUILD_SHARED_LIBS:BOOL=ON
-DTPL_BLAS_LIBRARIES:STRING=/usr/lib64/libblas.so
-DTPL_LAPACK_LIBRARIES:STRING=/usr/lib64/liblapack.so" > COMMON.config
echo "-DMPI_BASE_DIR:PATH=/home/jmwille/install/bin" > MPI_DEBUG.config

/home/jmwille/TrilinosTestHarness/Trilinos/checkin-test.py \
--make-options="-j6" \
--ctest-options="-j6" \
--ctest-timeout=300 \
$EXTRA_ARGS
```



## checkin-test.py: Example Driver Script

---

- If using a driver script, don't overwrite `.config` files manually
- Example driver scripts can be found in  
`sampleScripts/checkin-test-<machine>.sh`
- Run as (after symbolically linking into CHECKIN directory):  
`$ ./checkin-test-<mymachine>.sh --do-all --push`

A) Do local git commits

B) Run the checkin-test.py script:

```
$ ./checkin-test-mymachine.sh --do-all --push
```

C) Let the script run, results will appear in email

### Benefits:

- Documents a bullet-proof process for configuring, building, and testing Trilinos
- Does the VC commands to do a safe global checkin (ease git transition)
- Clear mind to work on something else (go home?) - results later in email
- Good sanity check at the end of stressful development effort
- Great for students and new developers
- Spend less time driving the checkin process / mitigate risk of overlooking an important build (ex serial)



## checkin-test.py: Log files

---

Directory Structure for auto-generated log files

CHECKIN/

  checkin-test.out

  update.out

MPI\_DEBUG/

  configure.out

  make.out

  ctest.out

SERIAL\_RELEASE/

  ...

See log files while configure, build, or test is being run:

```
$ tail -f MPI_DEBUG/make.out
```



## Speeding up Pre-Checkin Testing

---

- Slower to run solid checkin tests manually than it is using checkin-test.py
- Better approach: Make the script run faster, or customize (trim) testing
  - Speed up testing (all safe):
    - Checkin from fast remote workstation (easier with git)
    - Keep private development and checkin builds separate
    - Enabled shared libraries (`-DBUILD_SHARED_LIBS:BOOL=ON`)
    - Keep the CHECKIN builds up to date (through crontab or manually)
    - Fire off the script before you go home (latency hiding, not true speedup)
      - Can commit several times per day, and push once

- Be specific about what is tested (appropriate in certain situations):
  - Disallow enabling all packages (`--enable-all-packages=off`)
    - Example: When `cmake/TrilinosPackages.cmake` changes
  - Disable forward packages (`--no-enable-fwd-packages`)
    - Example: Only tests in the package have changed
    - Example: Good unit tests and minimal changes
  - Disabling specific downstream packages (`--disable-packages=P1,...`)
  - Enabling only specific packages (`--enable-packages=P1,...`)
    - Example: Only test a few packages  
`--enable-all-packages=off --enable-packages=Tpetra,Belos`



## checkin-test.py Script: Advanced Options

---

- **--extra-builds=BUILD1,...,BUILDN** Run extra user-defined build cases
  - Example: Test Secondary Stable Code and TPLs

```
$ echo "-DTPL_ENABLE_SCOTCH:BOOL=ON" >> WITH_SCOTCH.config
$ ./checkin-test-mymachine.sh --extra-builds=WITH_SCOTCH --do-all
```
  - MPI must be turned on explicitly for custom MPI builds.
  - Please run standard builds also (when possible)
- **--without-default-builds** Do only custom builds, please only use if necessary
- **--without-serial-release** Do only MPI\_DEBUG build
- **--show-defaults** Display default option values
- **--wipe-clean** Blow existing build directories and build/test results for specified builds.



## checkin-test.py Script: Advanced Options

---

- **--push** Can be used to push after a test run is complete. Replace **--do-all** (and **--commit** if previously used) with **--push**
- **--send-email-to-on-push=** List of comma-separated email addresses (lists?) to send email notification to on a successful push. To turn off notification, use **--send-email-to-on-push=""**.
- **--send-email-to=** List of comma-separated email addresses to send email notification to after every build/test case finishes and at the end for an overall summary and push status. By default, this is the email address returned by **eg config --get user.email**. To turn off email notification, set **--send-email-to=""** and no email will be sent.
- **--force-commit-push** Force the local commit and/or push even if there are build/test errors.
  - WARNING: Only do this when you are certain that the errors are not caused by your code changes.



## checkin-test.py Script: Very Advanced Options

---

- **--execute-on-ready-to-push=COMMAND** A command to execute on successful execution and 'READY TO PUSH' status. Can be used to do a remote SSH invocation to a remote machine to do a remote pull/test/push after this machine finishes. See scripts with '[remote](#)' in the name for an example.
- **--extra-pull-from=machine:/base/dir/repo:master** Optional extra git pull '<repository>:<branch>' to merge in changes from after pulling in changes from 'origin'. This pull is only done if --pull is also specified
  - NOTE: Modifying commits on remote machine will make your clone on your development machine incompatible with the repository.
  - NOTE: use 'eg pull; eg rebase --against origin/master' not eg pull --rebase



## Pre-Checkin Testing: Final Thoughts

---

- All checkin-test.py options are documented:
  - <https://software.sandia.gov/trilinos/developer/checkin-test.help.out>
  - \$ checkin-test.py --help
- The script will perform 2 builds by default.
- Other custom builds can be added.
- The amount of testing is fully customizable, so please use the script.